功能：deCasteliau算法计算n次贝齐尔曲线p(t)参数为t那点处的jd阶导矢pt，若jd=0,pt=p(t)。
输入参数： n-顶点数减1即次数；m_aVertex-控制顶点，为受保护成员；t-参数值；jd-导矢阶数(若求曲线上的点，输入jd＝０)。
输出参数：pt-曲线上参数为t的点(当jd＝0)或jd阶导矢(当jd≥１)。

```
void GetBezierCuvDerivat(int n,int jd,double t,CPoint &pt)
{
	CArray<double,double> TemVertex_x;
	CArray<double,double> TemVertex_y;
	TemVertex_x.SetSize(n+1);
	TemVertex_y.SetSize(n+1);
	pt.x=0;
	pt.y=0;
	if(jd>n) return;
	if(n>=0&&jd<=n)
	{
		for(int i0=0;i0<=n;i0++)
		{
			TemVertex_x[i0]=m_aVertex[i0].x;
			TemVertex_y[i0]=m_aVertex[i0].y;
		}
		for(int i1=1;i1<=n-jd;i1++)
		{
			for(int j=0;j<=n-i1;j++)
			{
				TemVertex_x[j]=TemVertex_x[j]+t*(TemVertex_x[j+1]-TemVertex_x [j];
				TemVertex_y[j]=TemVertex_y[j]+t*(TemVertex_y[j+1]-TemVertex_y[j]);
			}
		}
	}
	int njd=1;
	if(jd!=0)
	{
		for(int ll=1;ll<=jd;ll++)
		{
			njd=njd*(n-ll+1);
			for(int j=0;j<=jd-ll;j++)
			{
				TemVertex_x[j]=TemVertex_x[j+1]-TemVertex_x[j];
				TemVertex_y[j]=TemVertex_y[j+1]-TemVertex_y[j];
			}
		}
	}
	pt.x=int(njd*TemVertex_x[0]);
	pt.y=int(njd*TemVertex_y[0]);
}
```